Sensorwebs for Pursuit-Evasion Game on Berkeley UAV / UGV Testbed

S^5 :Sastry, Simic, Sinopoli, Schenato, and Shaffert, with help of the BEAR gang, J. Hu, and J. Zhang

Electrical Engineering & Computer Sciences

University of California, Berkeley

Sub-problems for PEG

- **Sensing**
  - **Navigation sensors -> *Self-localization***
  - **Detection of objects of interest**
- **Framework for communication and data flow**

SENSOR NETWORKS

- **Map building of environments and evaders**
  - **How to incorporate sensed data into agents' *belief states***
    → probability distribution over the state space of the world
    (I.e. possible configuration of locations of agents and obstacles)
  - **How to update belief states**
- **Strategy planning**
  - **Computation of pursuit policy**
    → mapping from the belief state to the action space
- **Control / Action**

Localization & Map Building

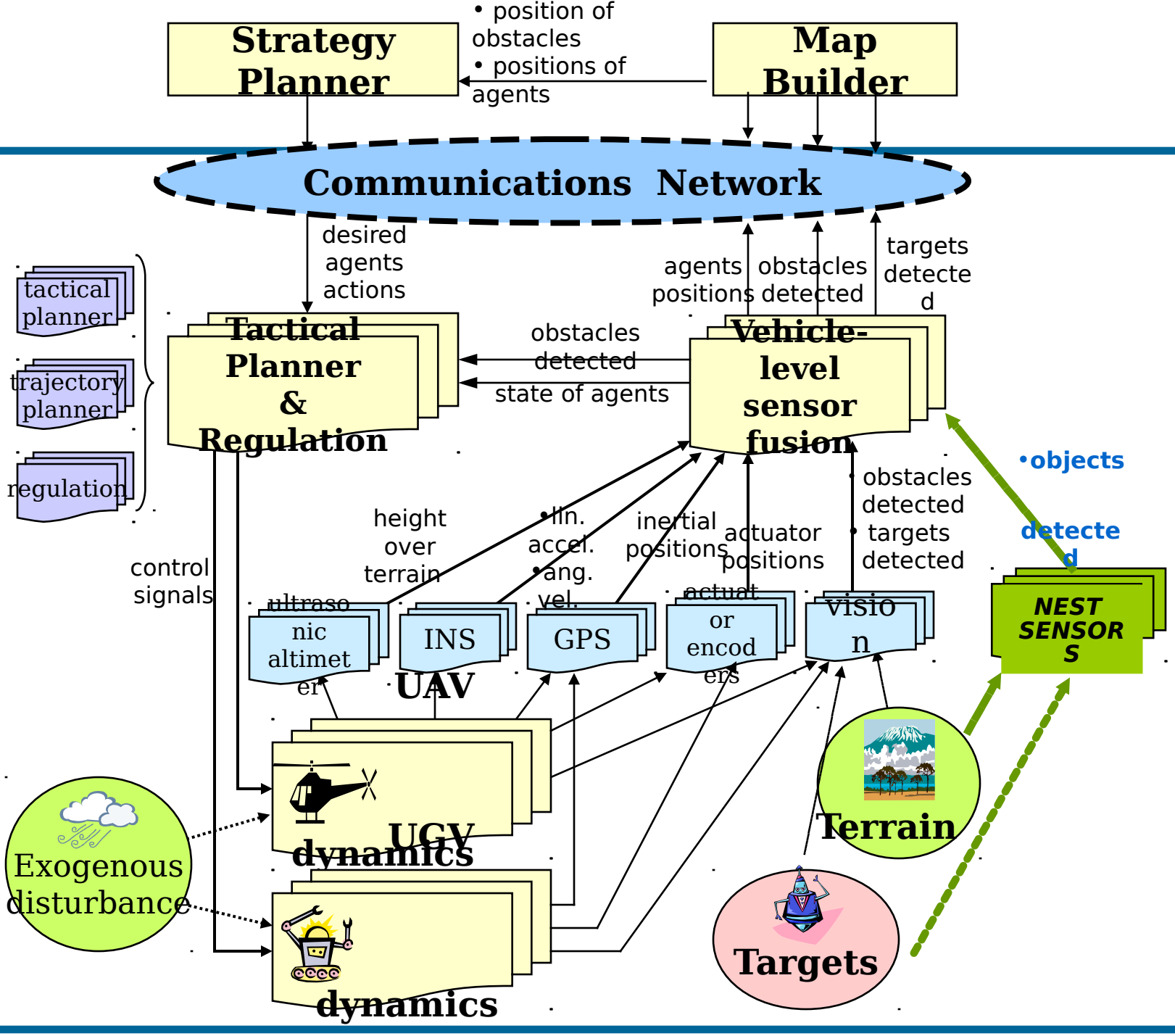
- **Localization : updating agent's position relative to the environment**
- **Map building: updating object locations relative to the agent's position or to the environment**

- **They can benefit from different techniques, e.g.,**

Occupancy-based : well-suited to path planning, navigation, and obstacle avoidance, expensive algorithms (e.g. pattern matching) required for localization

Beacon-based : successful to localization

Fails in cluttered environment, unknown types of objects


Strategy Planner
• position of obstacles
• positions of agents
Map Builder
Communications Network
desired agents actions
agents positions
obstacles detected
targets detected
tactical planner
trajectory planner
regulation
Tactical Planner & Regulation
obstacles detected
state of agents
Vehicle-level sensor fusion
control signals
height over terrain
•lin. accel.
•ang. vel.
inertial positions
actuator positions
• obstacles detected
• targets detected
•objects detected
ultrasonic altimeter
INS
GPS
actuator encoders
vision
NEST SENSORS
UAV
UGV dynamics
dynamics
Exogenous disturbance
Terrain
Targets

Optimal Pursuit Policy

- **Performance measure : capture time**

$$\mathbf{T}^* := \min\left\{ \tau \geq 1 : \mathbf{Y}_\tau \notin Y_\tau^{\neg cap} \right\}$$

where $Y_\tau^{\neg cap}$ is the set of all $\{y(1)...y(\tau)\}$,

associated with an evader not being captured up to τ

- **Optimal policy μ minimizes the cost**

$$J_\mu := \mathrm{E}_\mu [\mathbf{T}^*]$$

Optimal Pursuit Policy

- **cost-to-go for policy μ, when the pursuers start with $\mathbf{Y}_t = Y_t$**

$$V_\mu(Y_t, \eta^t) \triangleq \sum_{s \in \mathcal{S}} E_\mu[\min\{\tau - t \geq 1 : \mathbf{Y}_\tau \notin \mathcal{Y}_\tau^{\neg cap}\} \mid \mathbf{s}(t) = s, \mathbf{Y}_t = Y_t]\pi_s^t$$

n η^t for the

$$= 1 + \sum_{\{y_{t+1} \in \mathcal{Y} : Y_{t+1} \in \mathcal{Y}_{t+1}^{\neg cap}\}} \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} V_\mu(Y_{t+1}, \pi^{t+1}) O(y_{t+1}, s', a_t) T(s', s, a_t) \pi_{(s)}^t$$

- **cost of policy μ**

$$J_\mu := \sum_{y \in \mathsf{Y}} \mathrm{E}_\mu \left[\min\left\{ \tau \geq t+1 : \mathbf{Y}_\tau \notin \mathsf{Y}_\tau^{\neg fnd} \right\} \middle| \mathbf{y}(1) = y \right] \mathrm{P}(\mathbf{y}(1))$$

$$= \sum_{y \in \mathsf{Y}} V_\mu(\{y\}, \pi_\mu(\{y\})) \, \mathrm{P}(\mathbf{y}(1))$$

Persistent pursuit policies

- **Optimization using dynamic programming is computationally intensive.**
- **Persistent pursuit policy** g

$$\mathrm{P}_g(\mathbf{T}^* = t \mid \mathbf{T}^* \geq t) \geq \varepsilon > 0$$

$$\rightarrow \quad E_g[\mathbf{T}^*] = \sum_{t=1}^{\infty} t\mathrm{P}_g(\mathbf{T}^* = t) \leq \varepsilon^{-1}$$

- **Persistent pursuit policy** g **with a period** T

$$\mathrm{P}_g(\mathbf{T}^* \in \{t, \ldots, t+T-1\} \mid \mathbf{T}^* \geq t) \geq \varepsilon > 0$$

$$\rightarrow \quad E_g[\mathbf{T}^*] = \sum_{t=1}^{\infty} t\mathrm{P}_g(\mathbf{T}^* = t) \leq T\varepsilon^{-1}$$

Pursuit Policies

- **Greedy Policy**
  - **Pursuer moves to the adjacent cell with the highest probability of having an evader over all maps**
  - **Desired location and heading for the pursuer are given by**

$$x_d(t+1) = \underset{x_k \in \mathcal{X}_{p_k}}{\operatorname{argmax}} \sum_{k=1}^{n_p} p_e(x_k, t+1 \mid Y_t)$$

$$\psi_d(t+1) = \underset{\psi \in \Psi_{p_k}}{\operatorname{argmin}} \mid \angle(x_d(t+1) - v_k(t)) - \psi \mid^2,$$

where $\mathcal{X}_{p_k}$ and Ψ_{p_k} are one-step reachability region of pursuer k, and n_p is the number of pursuers.

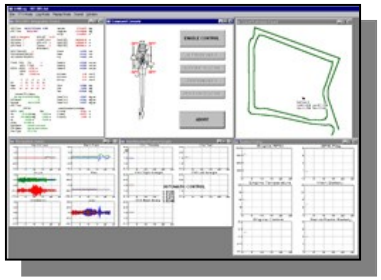
Pursuit Policies

- **Global-Max Policy**
  - **Pursuer moves towards the place with the highest probability of having an evader in the map**

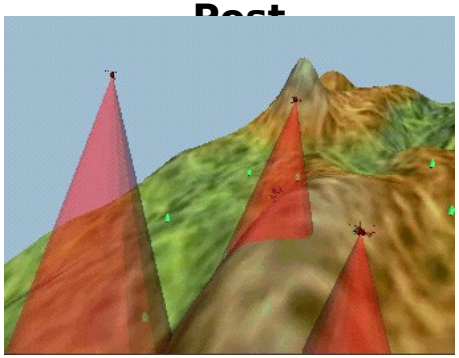
$$x_d(t+1) = \underset{x' \in \mathcal{X}_{p_k}}{\operatorname{argmin}} \; d\left(x', \underset{x \in \mathcal{X}}{\operatorname{argmax}} \; \frac{p_e(x, t+1 \mid Y_t)}{d(x, v_k(t))}\right)$$

$$\psi_d(t+1) = \underset{\psi \in \Psi_{p_k}}{\operatorname{argmin}} \; \angle(x_d(t+1) - v_k(t)) - \psi \; |^2 \; .$$

Pursuit-Evasion Game Experiment Setup



Ground Command Post



Waypoint Command

Current Position, Vehicle Stats

Evader location detected by Vision system

Current Position, Vehicle Stats

Pursuer: UAV

Evader: UGV

Experimental Results: Pursuit-Evasion Games with 4UGVs and 1 UAV (Spring' 01)

Issues in current setup

- Current BEAR Framework for PEG
  - Navigation sensors(INS, GPS, ultrasonic sensor…) for localization
  - Ultrasonic sensor for obstacle avoidance
  - Vision-based detection for moving targets (enemy)
  - Occupancy-based map building for planning
- Potential Issues for real-world PEG
  - GPS jamming, unbounded error of INS, noisy ultrasonic sensors
  - Computer vision algorithms are expensive
  - Cameras have small range
  - Unmanned vehicles are expensive

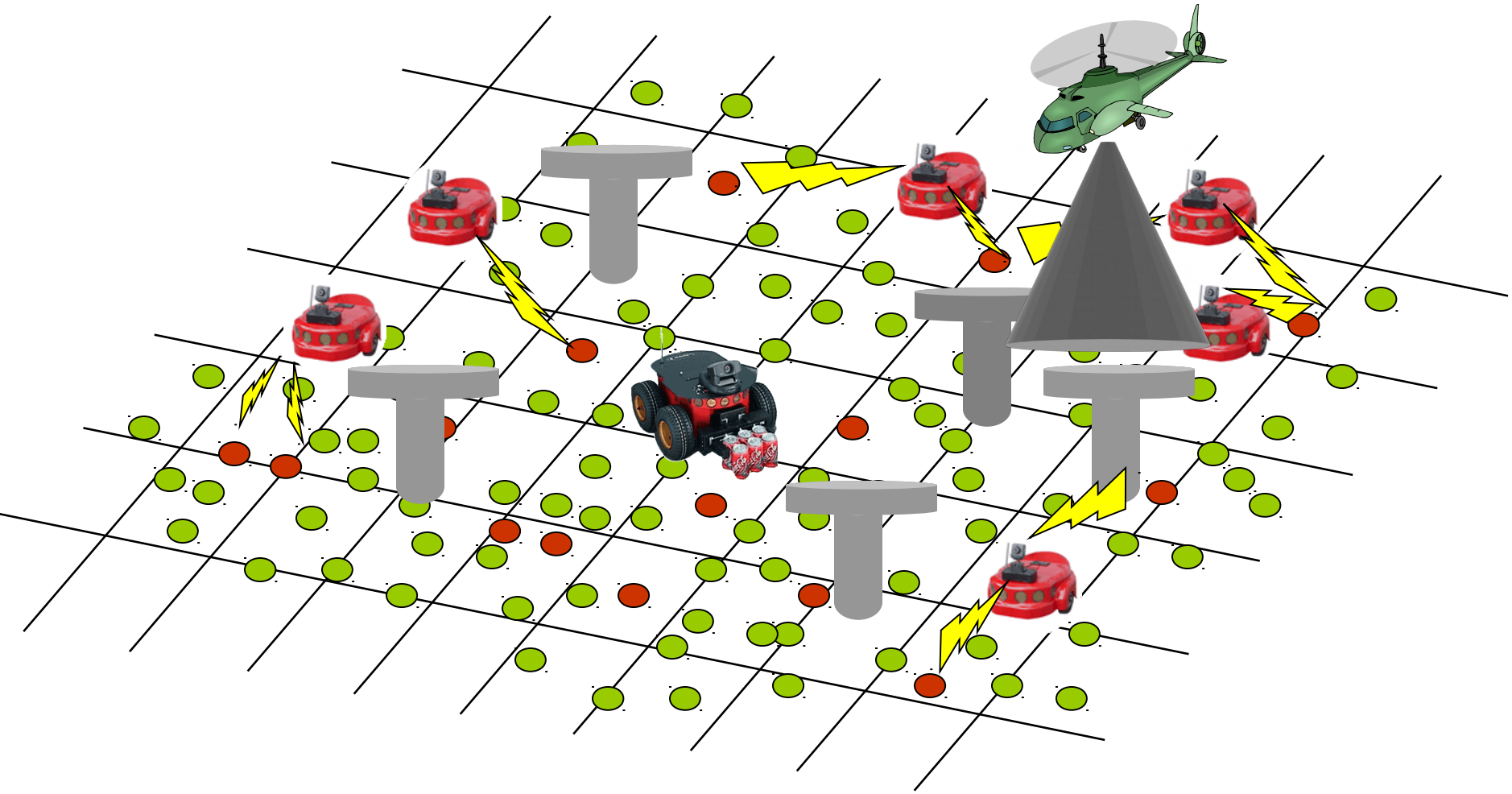
    → It is unrealistic to employ many number of unmanned vehicles to cover a large region to be monitored.

    → Static optimal placement of unmanned vehicles for cooperative observations are already difficult (e.g. art-gallery or vertex-cover problems).

The role of a sensor network

- **Provide complete monitoring of the environment, overcoming the limited sensing range of on board sensors**
- **Relay secure information to the pursuers to design and implement an optimal pursue strategy**
- **Possibly provide guidance to pursuers, when GPS or other navigation sensors may fail**

Distributed Pursuit Evasion Games (DPEGs)



* Robot pictures from ActivMedia webs

Toward playing PEGs with sensor network

- **Leverage the work already demonstrated by BEAR team**
- **Develop a tracking algorithm for the SN**
- **Integrate Sensor Network (SN) in the most seamless way by identifying the exchange of information between SN and ground or/and aerial pursuers**
- **Develop clustering algorithms for data aggregation**
- **Develop application specific communication protocols**

Components needed for DPEGs

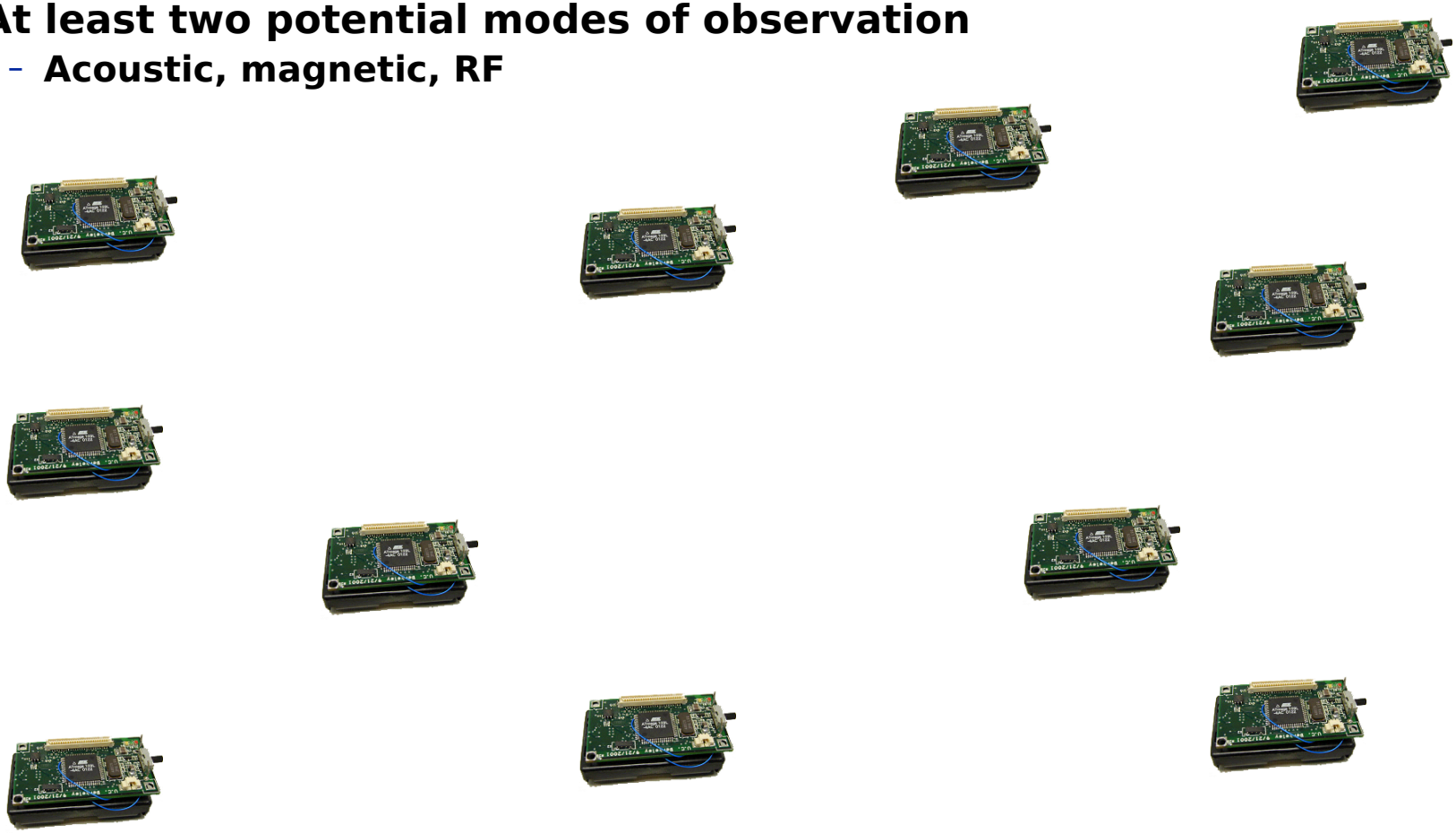
- Time synchronization
- Self-organized dissemination and processing
- Local coordinate system
- Triggered Reconfiguration
- Identification
- Target localization
- Tracking

Platform

- **Large number of MICA constrained wireless nodes**
  - **two mode of sensing (acoustic and magnetic or vibration)**
  - **limited radio range**
  - **TinyOS event-driven OS structure**
  - **limited energy reserves**
- **Small number of more powerful nodes**
  - **bridge short-range RF to long range communication**
  - **processing and storage capabilities**
- **High powered surveillance cameras**
  - **associated with power nodes**
  - **video capability – detailed, but not covering entire space**
  - **pan and zoom**

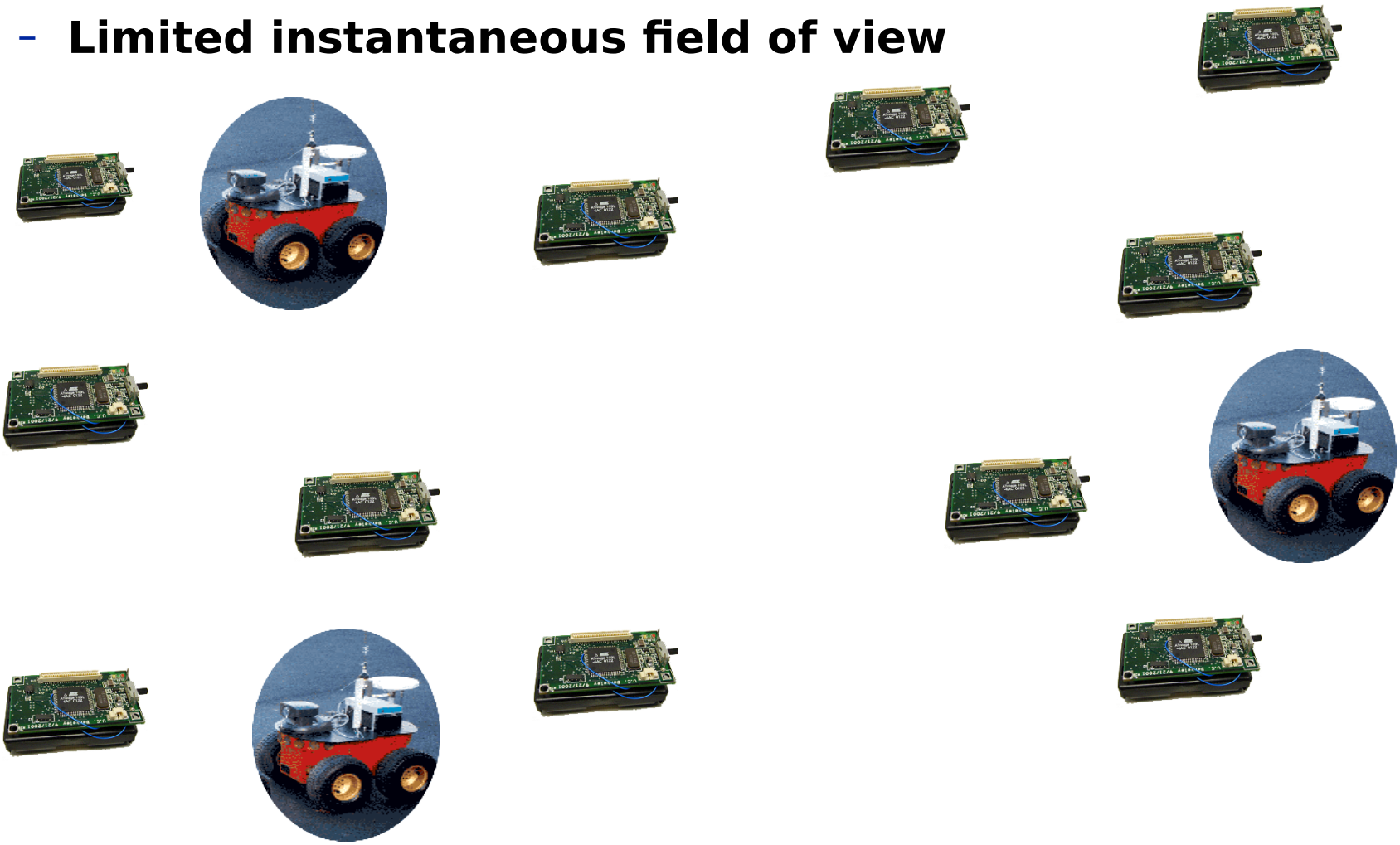
1. Field of wireless sensor nodes

- **Ad hoc, rather than engineered placement**
- **At least two potential modes of observation**
  - **Acoustic, magnetic, RF**

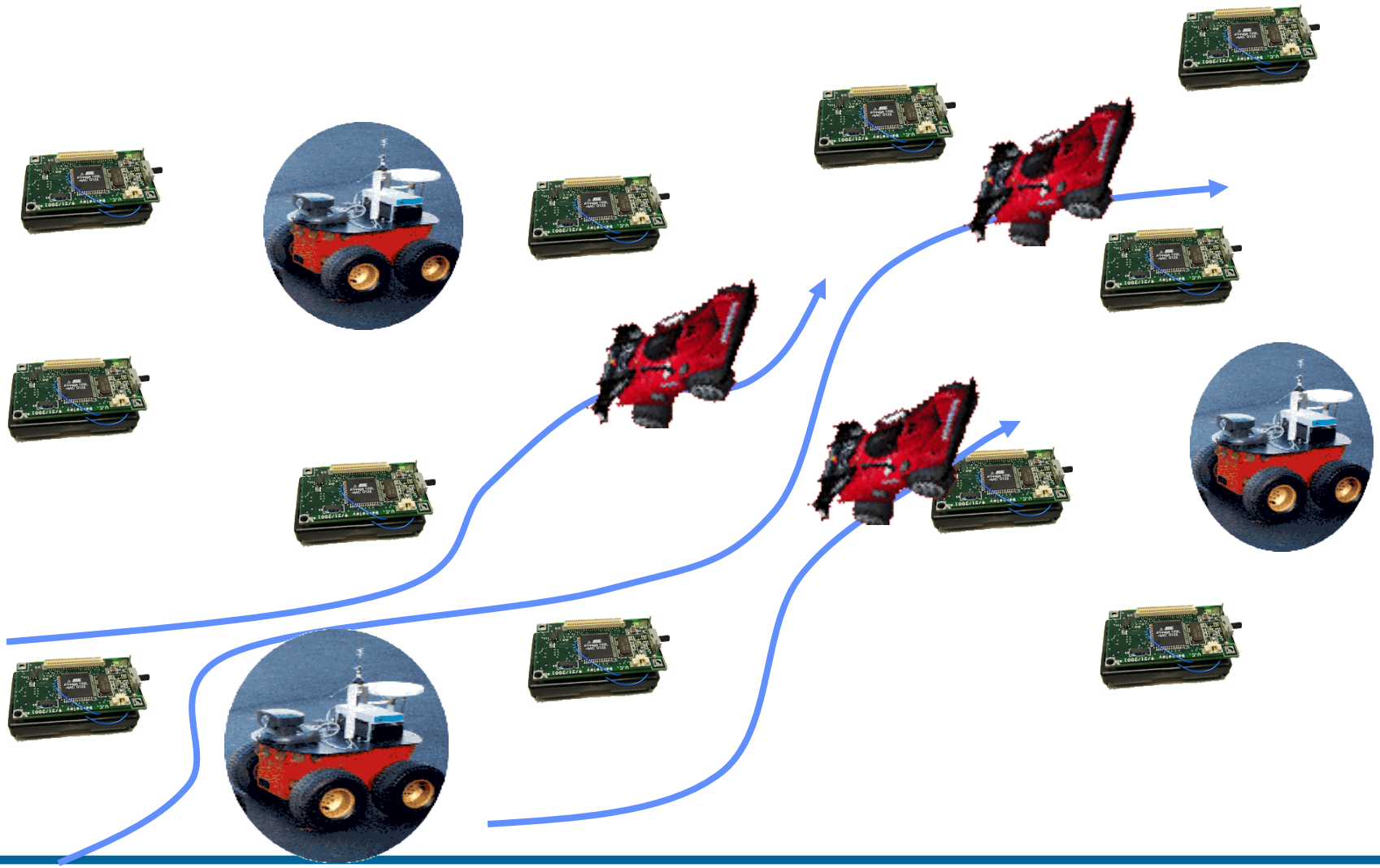


2. Subset of more powerful assets

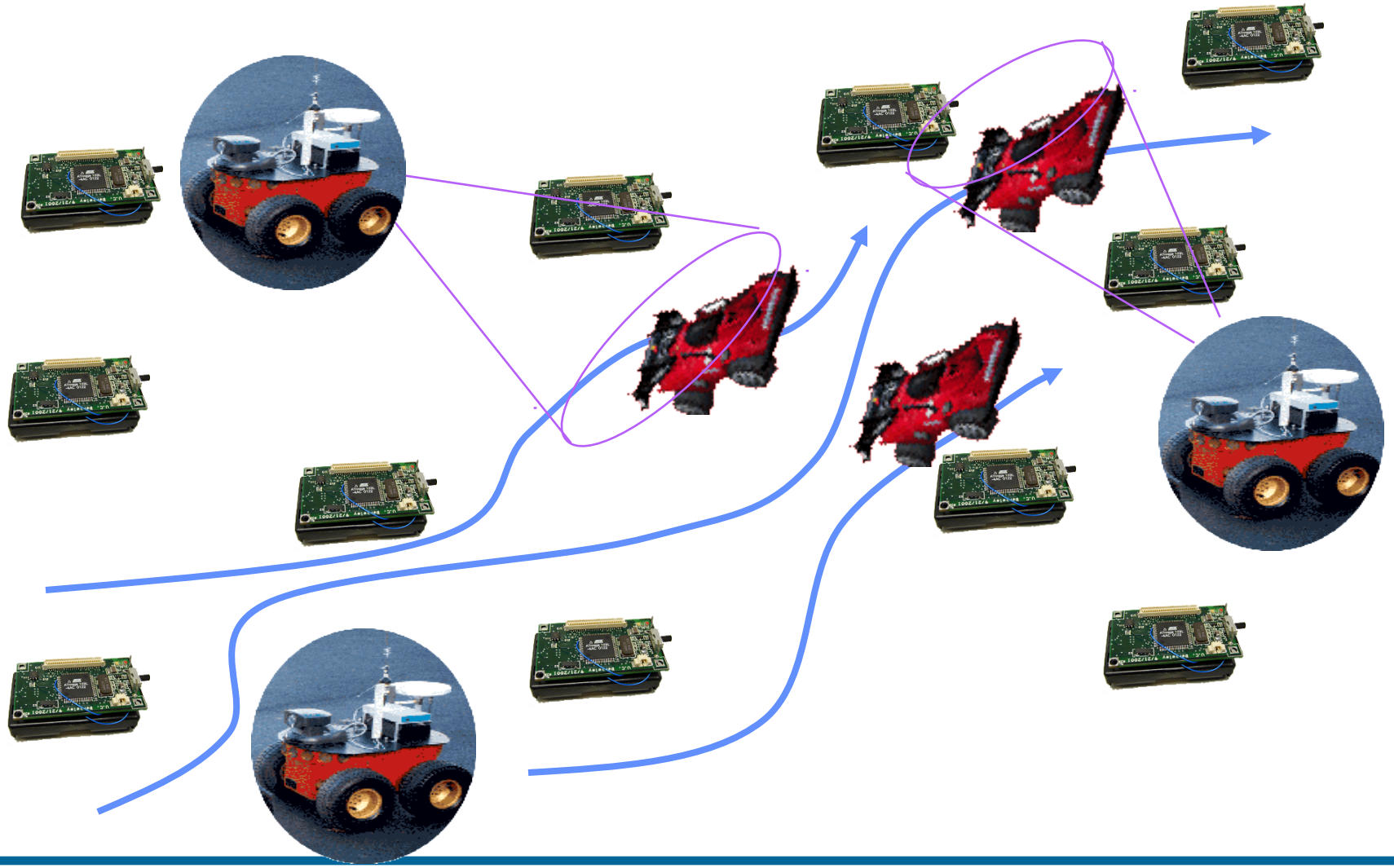
- **Gateway nodes with pan-tilt camera**
  - **Limited instantaneous field of view**



3. Set of objects moving through



4. Track a distinguished object



Many interesting problems arise from this set up

- Targeting of the cameras so as to have objects of interest in the field of view
- Collaborate between field of nodes and platform to perform ranging and localization to create coordinate system
- Building of a routing structures between field nodes and higher-level resources
- Targeting of high-level assets
- Sensors guide video assets in real time
- Video assets refine sensor-based estimate
- Network resources focused on region of importance

Abstraction of Sensorwebs

- **Properties of general sensor nodes are described by**
  - **sensing range, confidence on the sensed data**
  - **memory, computation capability**
  - **Clock skew**
  - **Communication range, bandwidth, time delay, transmission loss**
  - **broadcasting methods (periodic or event-based)**
  - **And more…**

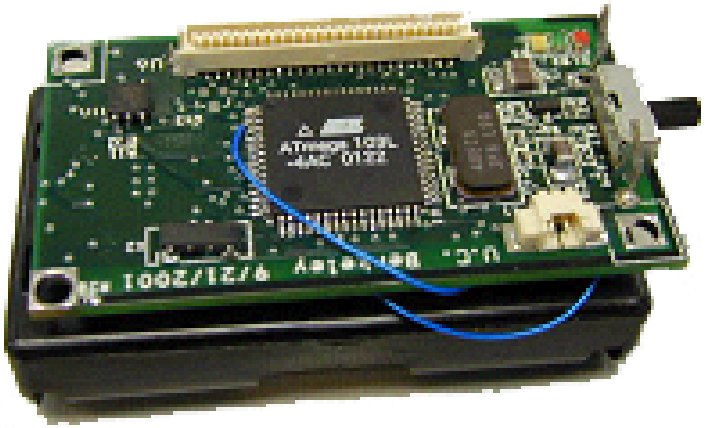
- **To apply sensor nodes for the experiments with BEAR platform,**

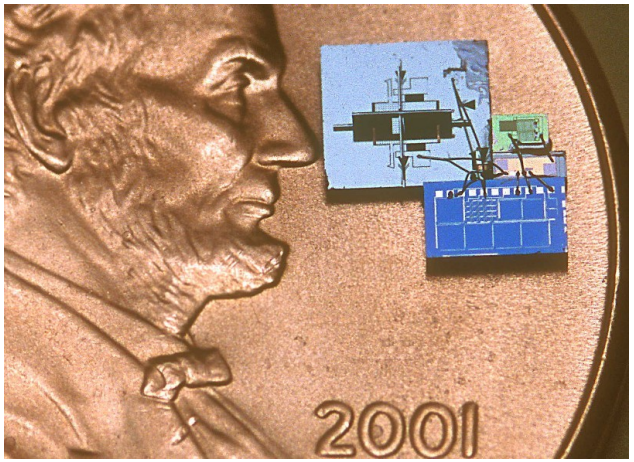
  introduce super-nodes (or gateways), which can
  - **gather information from sub-nodes**

    (filtering or fusion of the data from sub-nodes for partial map building)
  - **communicate with UAV/UGVs**

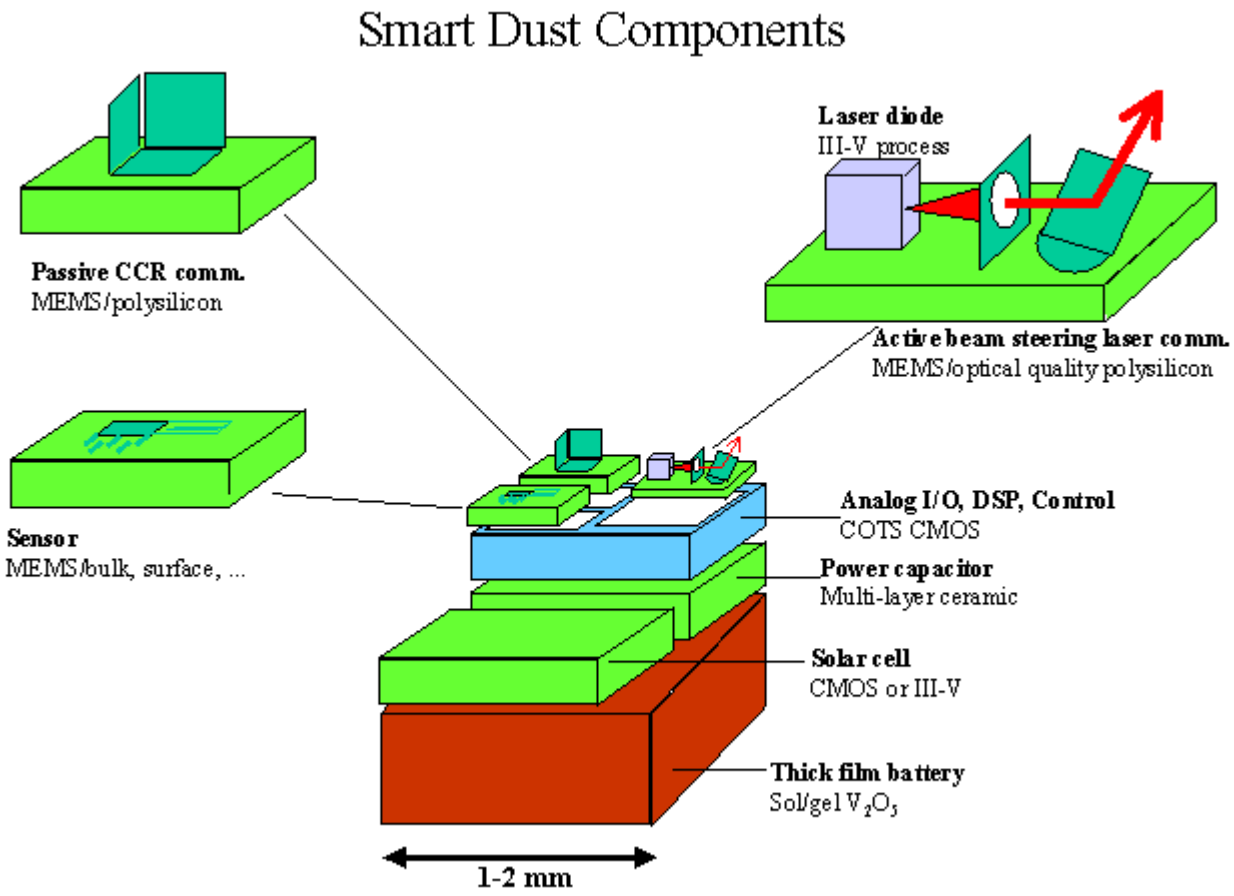
Smart Dust, Dot Motes, MICA Motes

Dot motes, MICA motes and smart dust





August ’01 Goal

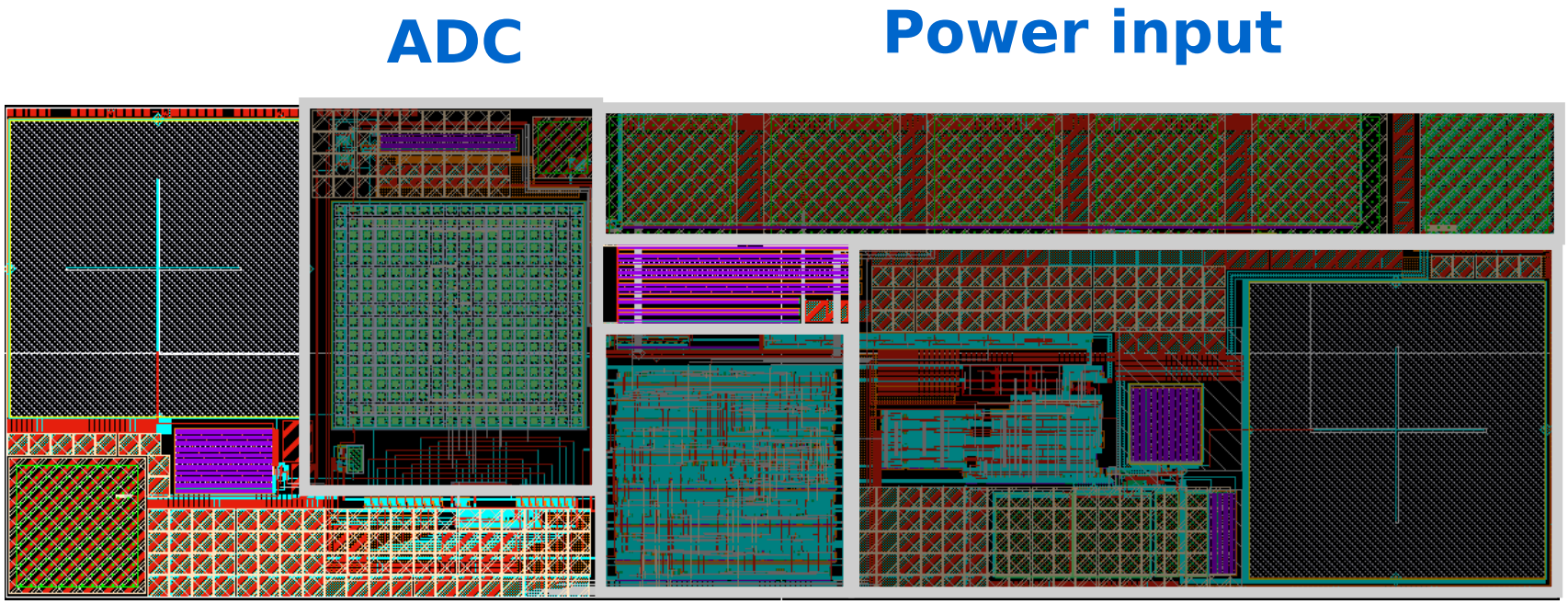


Power and Energy

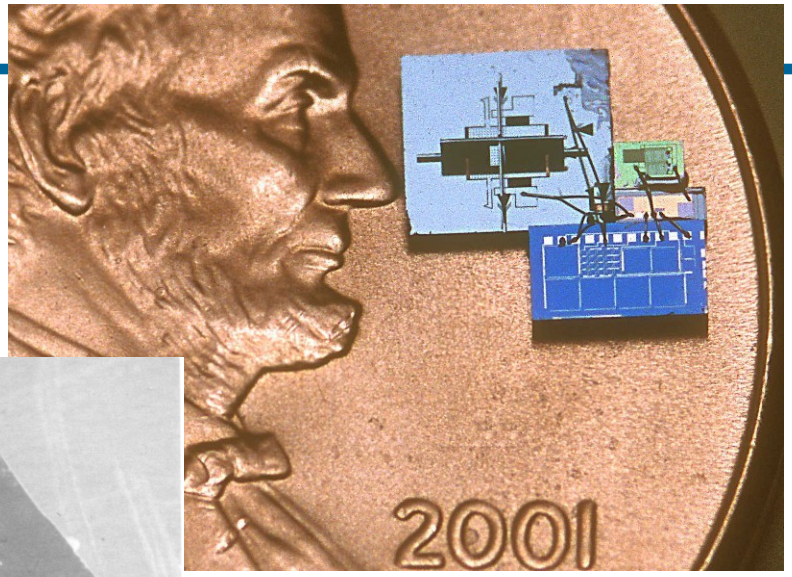
- **Sources**
  - **Solar cells ~0.1mW/mm^2, ~1J/day/mm^2**
  - **Combustion/Thermopiles**
  - **Vibration**
- **Storage**
  - **Batteries ~1 J/mm^3**
  - **Capacitors ~0.01 J/mm^3**
- **Usage**
  - **Digital computation: nJ/instruction**
  - **Analog circuitry: nJ/sample**
  - **Communication: nJ/bit**

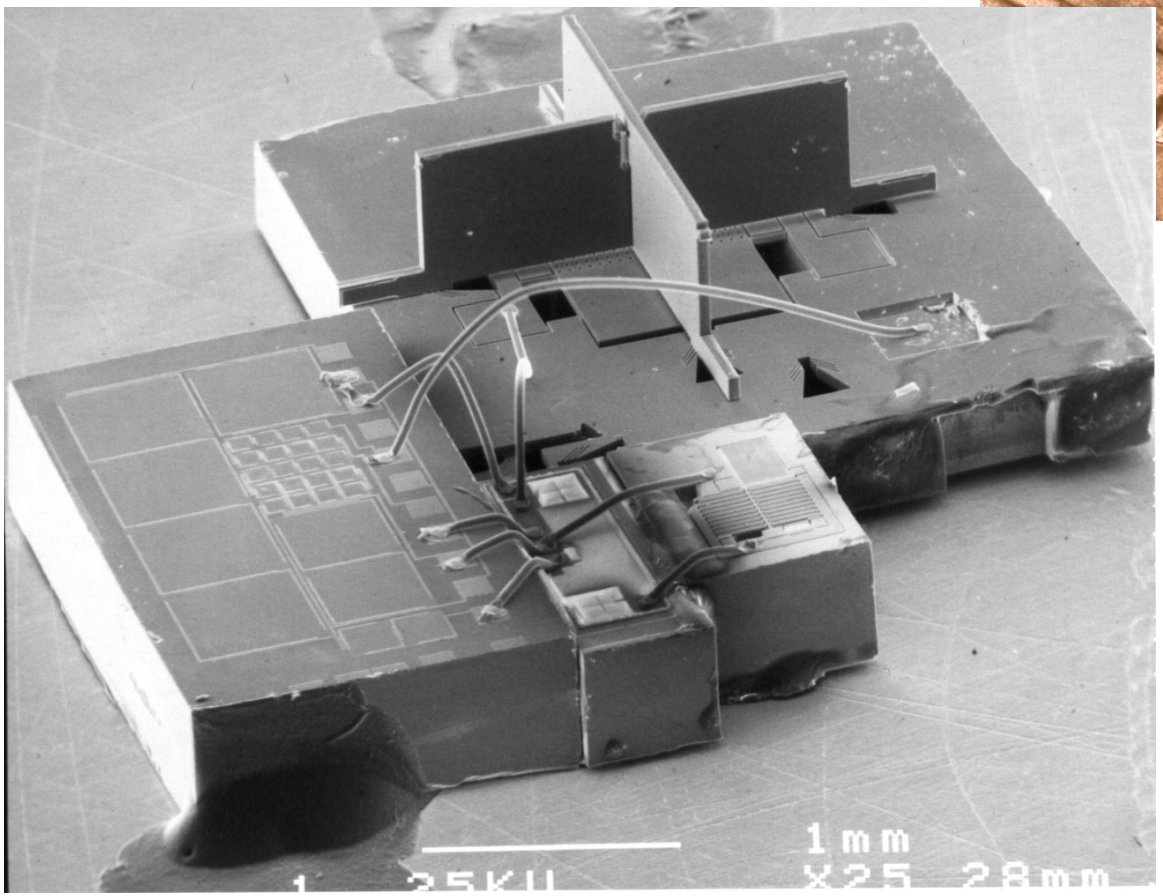
½ of first real attempt

Warneke, Leibowitz, Scott, Boser



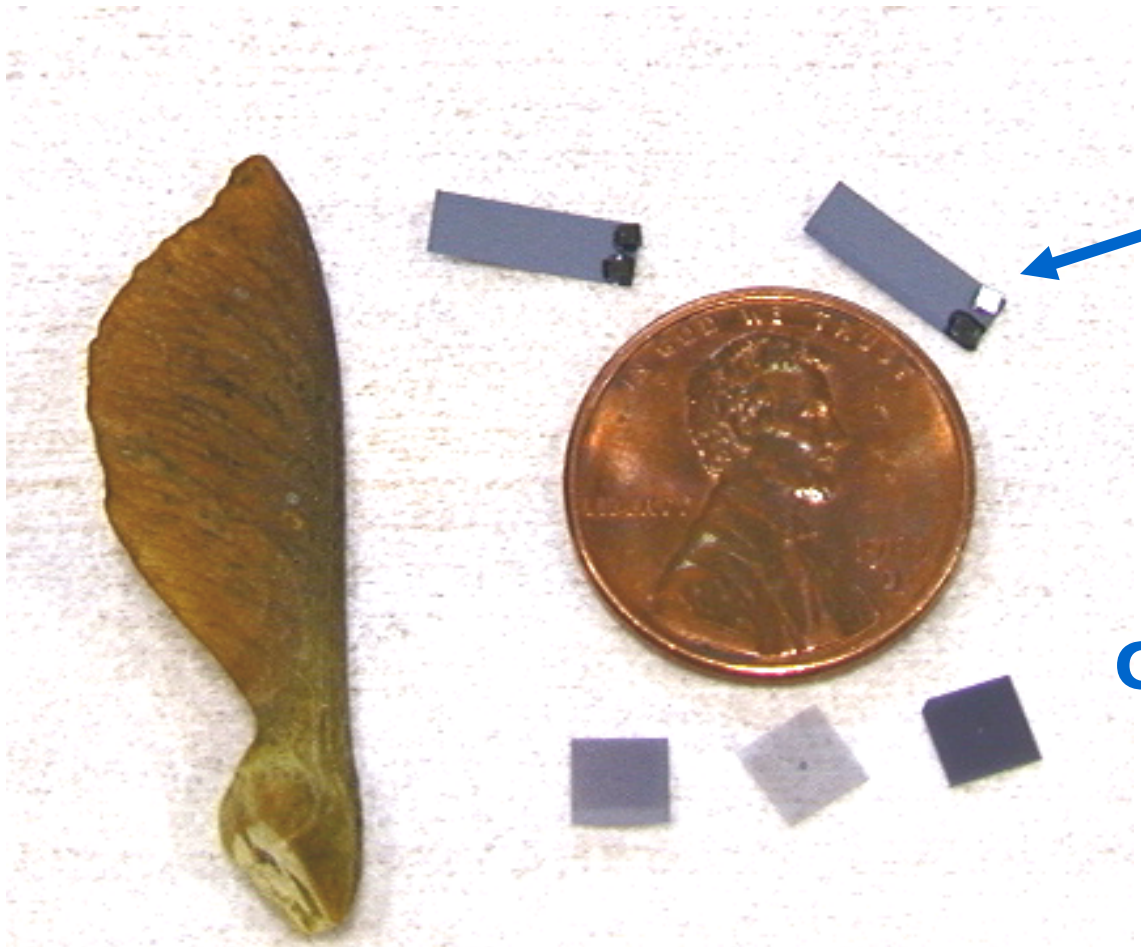
Dust Mock-up





Dust Delivery

- **Silicon maple seeds, dandelions**



1mm^3

Solar power,
Gossamer wings

Sensorwebs: The Abstracted Setting

- **Deployment:** N sensor nodes are randomly scattered in an area of operations, Q; each node has sensing radius R and communication radius r.

- **Network:** They form an *ad hoc communication network* – two nodes can communicate if they are less than r meters apart, but there is no a priori routing protocol.

- **Fundamental problems underlying PGE:**
  - ***Localization of nodes***
  - ***Tracking of moving objects***
  - ***Environmental monitoring***
  - ***Map building***

Localization

- **Problem formulation:** *given that some (say K) nodes in a Sensorweb know their positions in a fixed coordinate system, compute the positions of the remaining N - K nodes.*

- **Goal:** *design scalable distributed algorithms for localization.*

- **Why distributed?**
  - Long-range, multi-hop communication with a central computing unit is expensive: trade-off between computation and communication
  - Each mote has an on-board computer equipped with ***Tiny OS,*** capable of performing basic operations
  - Decentralized, collaborative approach can lead to faster, more energy efficient and more robust algorithms

Approaches to Localization

- **Basic observation:**
  - If an unknown sensor can receive communication signals from a nearby beacon or node, it lies in a disc centered at that beacon/node with radius *r.*
  - If it receives position information from *m* nearby beacons, it lies in the intersection of these *m* discs.

- **Approaches:**
  - ***Ellipsoidal:*** the intersection of discs is outer-approximated by an ellipsoid
  - ***Polytope*****:** the intersection of discs is outer-approximated by a polytope
  - ***Discretized*****:** the area of operations is divided into cells by a grid and discs are approximated by squares

- **Distributed aspect:** Every sensor performs its own position estimation using its own computational power, and the estimated position is stored in local memory

Discrete approach

- **Basic assumptions:**
  - Area of operations Q is a square.
  - Q is divided by a regular grid into n^2 ***cells***.
  - Two nodes can communicate if they are less than r cells apart.
  - K nodes know their positions.

- **Goals:**
  - Given an unknown node S, compute the cell in which it lies.
  - Compute the expected size of the estimate.
  - Compute the probability that the estimate is ***one cell*** in size (I.e., perfect).
  - Given a desired degree of accuracy, choose optimal network parameters.

- **Advantages:**
  - The approach allows for ***analytical*** estimates.
  - Implementable in Tiny OS.

Localization procedure, I

- S = an unknown node
- $S_1,\ldots,S_m$ = the known neighbors of S
- B_i= communication range of S_i
- Then S belongs to

$$L(S) = B_1 \text{Å} \ldots \text{Å} B_m.$$

- Note: it is easy to compute the intersection of squares – can be done even with limited computational power of Rene motes.
- Each S performs the following steps:

Step 1: *Gather positions of known neighbors.*

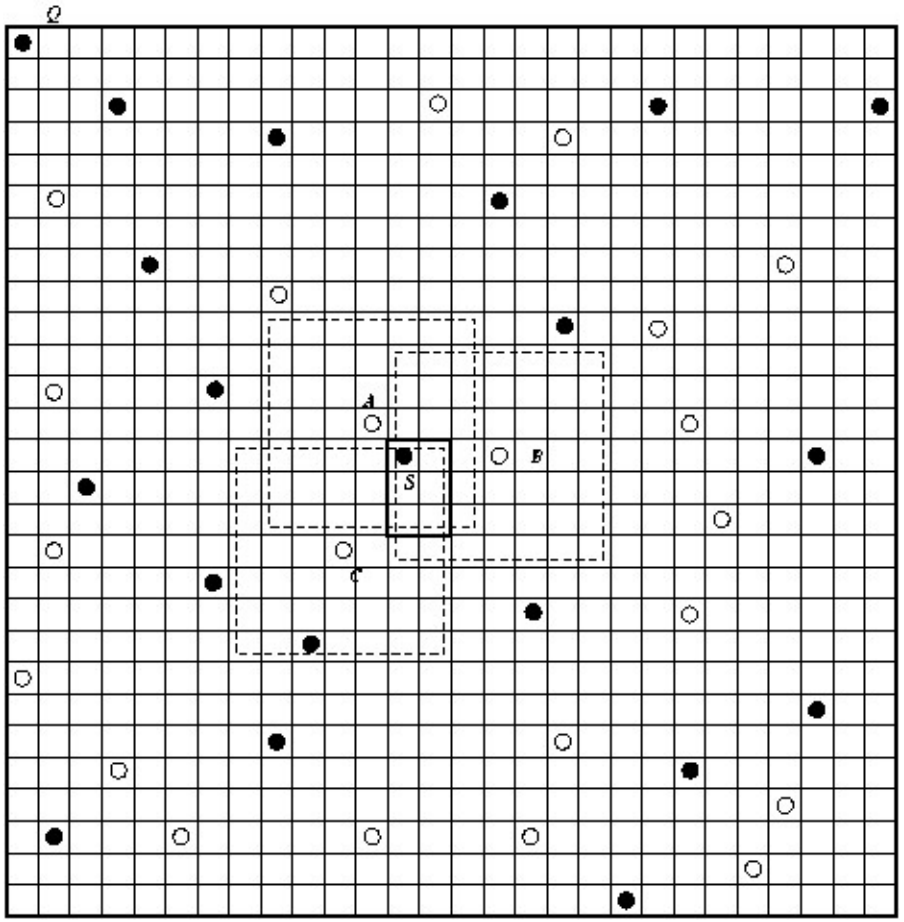
Step 2: *Compute $L(S)$ given above.*

Localization procedure, II

Unknown node S with known neighbors A,B,C.

Communication ranges of are in dashed lines.

$L(S)$ is the solid rectangle.



Distributed algorithm for localization

Each unknown node S executes the following algorithm LOC_S:

Step 1: *INITIALIZE the estimate:* $L(S) = Q$.

Step 2: *SEND "Hello, can you hear me?"*

- Each known neighbor sends back (1,a,b), where (a,b) is its position, each unknown neighbor sends (0,0,0).

Step 3: *For each received message* (1,a,b), *UPDATE the estimate:* $L(S) := L(S) \text{Å} [a - r,a+r] \times [b - r,b+r]$.

- Note: $[a - r,a+r] \times [b - r,b+r]$ is the communication range of the node (a,b).

Step 4: *STOP when all the messages have been received. The position estimate is* $L(S)$.

Analytical estimates

- Suppose S is an unknown node randomly picked at a distance of more than r cells from the boundary of Q. If the total number of known nodes is K, then the expected value of A_S (the size of the position estimate $L(S)$) is

$$\boldsymbol{E(A_S) = 1 + 4 \sum_{k=1}^{2r} \sum_{l=1}^{2r+1} \{1 - [(2r+1)^2 - kl]/n^2\}^K}$$

- *Observe*: $\boldsymbol{E(A_S) \to 1}$ **cell**, as $K \to \infty$,

  where one cell corresponds to the *perfect* estimate.

Analytical estimates, cont'd

- Assume there is the total of K known nodes. Let S be randomly picked at least r cells away from the boundary of Q. Denote by H_S the number of known neighbors of S. Then the conditional probability that A_S equals one given $H_S = m$ is

$$\boldsymbol{P(A_S = 1 | H_S = m) = [1 - (2r/n)^m]^4.}$$

- The probability that $A_S = 1$ is

$$\boldsymbol{P(A_S = 1) = \sum_{m=0}^{K} \binom{K}{m} p^m q^{K-m} [1 - (2r/n)^m]^4},$$

where $p = (2r + 1)^2/n^2$ is the area of the communication region, and $q = 1 - p$.

Choosing optimal network parameters, I

- Suppose we want the estimate to be “almost perfect”, $|E(A_S) - 1| < \varepsilon$.

To achieve this, we need $K \geq K_\varepsilon(n,r)$, where $K_\varepsilon(n,r)$ can be computed. This allows us to choose the right K given ε.

- There is a number $\delta_\varepsilon(r)$ such that, given ε, the density satisfies

  $K_\varepsilon(n,r)/n^2 \leq \delta_\varepsilon(r)$.

  for all n. We call $\delta_\varepsilon(r)$ the *critical density*.

- The **average complexity** of LOC_S which achieves $|E(A_S) - 1| < \varepsilon$ or $P(A_S = 1) > 1 - \varepsilon$ is $O(\log(1/\varepsilon))$.

Choosing optimal network parameters, II

- How do we ensure that K (the total number of known nodes) is *large enough*?
- We can:
  - Equip K nodes with ***GPS***, or
  - Prior to deployment, place ***beacons*** in Q. If they can localize every node which falls in some percentage, say α, the area of Q, then the expected value of the (in this setting) *random variable* K is

$$E(K) = N\alpha.$$

Therefore, by choosing N (the total number of nodes) large enough, we can make K sufficiently large.

Approach to tracking:

- **Design of tracking algorithm must be independent of the specific implementation of middleware such as:**
  - Synchronization
  - Localization
  - Communication protocols
  - Network preprocessing
- **Sensor network outputs:**
  - Position, velocity estimate of evader
  - Time stamp
  - Error bounds (variance) of position estimate

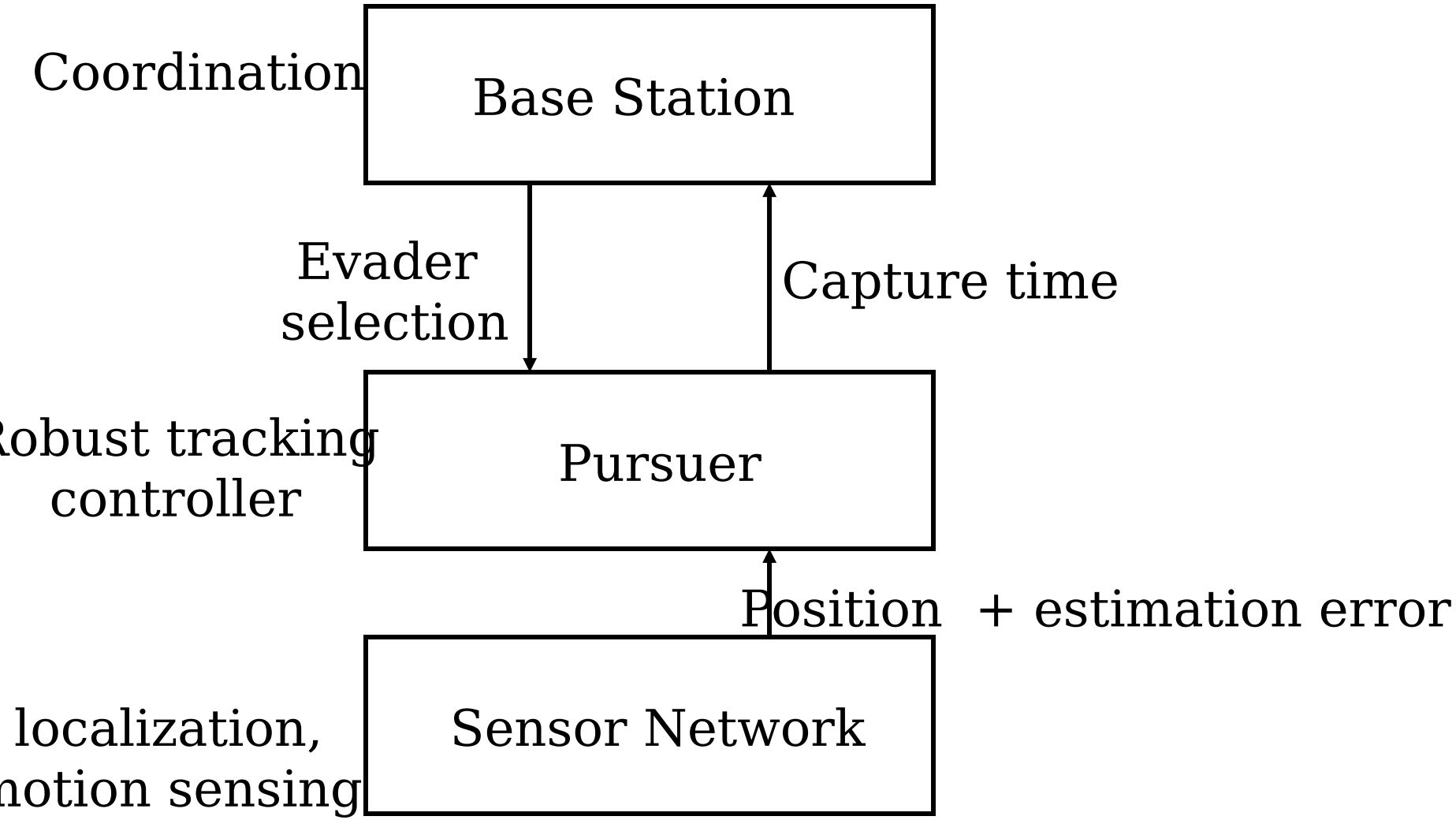
System parameters:

- **Sensor network features:**
  - Average nodes distanc$d$:
  - Sampling period:T
  - Evader position estimation error variance: σ_e
  - Estimation delay: τ
- **Evader features:** v_e
  - Maximum speed:
- **Pursuer features** v_p
  - Maximum sp$T_g$ed:
  - GPS period:

Objective:

- **Performance metrics:**
  - Average capture time T_c
  - Mean evader-pursuer distanc λ_m
- **GOAL:**
  - Design controller for the pursuer based on sensor network and GPS information
  - Estimate performance of controller as function of the network and evader features

Layered architecture: modular modeling



Problem formulation:

- **Position estimation layer:**
  - Position of evader(s) $p^e = (p^e_x, p^e_y)$
  - Position of pursuer(s) $p^p = (p^p_x, p^p_y)$
  - Estimated position of evad $\widehat{p}^e = (\widehat{p}^e_x, \widehat{p}^e_y)$
  - Evader estimation error σ_e
- **Network Outputs:**

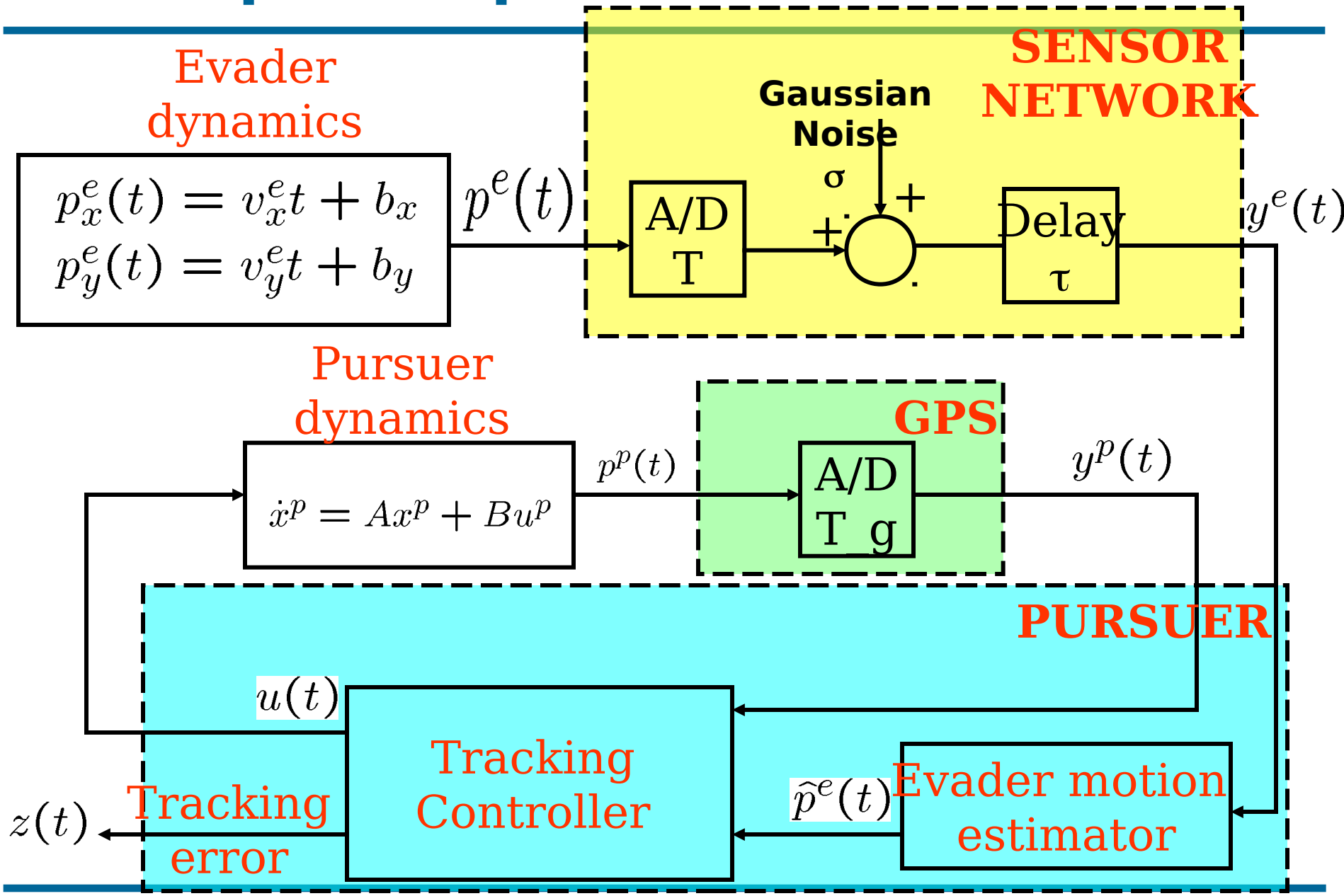
$$y^e(t) = \widehat{p}^e(t-\tau),\ ||\widehat{p}^e - p^e|| \sim \mathcal{N}(0, \sigma_e^2 \mathcal{I})$$

- **GPS output:** $y^p(t) = p^p(t)$

Simplified system dynamics:

- **Evader dynamics: constant velocity**
  - State: $x^e = (p_x^e, p_y^e, v_x^e, v_y^e)$
  - Evolution: $p_x^e(t) = v_x^e t + b_x$
    $p_y^e(t) = v_y^e t + b_y$
    (b_x, v_x^e, b_y, v_y^e) Unknown but constant
- **Pursuer dynamics: holonomic case**
  - State: $x^p = (p_x^p, p_y^p, v_x^p, v_y^p)$
  - Evolution: $\dot{v}_x^e(t) = -b\, v^e(t) + u_x^e$
    $\dot{v}_y^e(t) = -b\, v^e(t) + u_y^e$
    $|u_x| \leq b v_p, \quad |u_y| \leq b v_p$ Bounded input

State space representation:



Subproblems:

- **Evader motion estimator:**
  - Estimate (b_x, v_x^e, b_y, v_y^e) and their variances

    $(\sigma_{b_x}, \sigma_{v_x^e}, \sigma_{b_y}, \sigma_{v_y^e})$ using sensor network outputs.

- **Pursuer controller design:**
  - Design tracking controller such that

    $||z(t)|| \rightarrow 0 \qquad t \rightarrow \infty$

Evader Motion Estimator

- **On-line Least Square: Optimal**
  - Unknown motion parameters to be estimated:

$$\theta_x = \begin{bmatrix} v_x \\ b_x \end{bmatrix} \quad \theta_y = \begin{bmatrix} v_y \\ b_y \end{bmatrix}$$

  - Incoming data from sensor network:

$$(\hat{p}_x, \hat{p}_y, t) \qquad h = \begin{bmatrix} t \\ 1 \end{bmatrix}$$

2x2 Matrix

$$\theta_x^{(n+1)} = \theta_x^{(n)} + H_n^{-1}(\hat{p}_x - h^T\theta_x^{(n)})h$$
$$\theta_y^{(n+1)} = \theta_y^{(n)} + H_n^{-1}(\hat{p}_y - h^T\theta_y^{(n)})h$$

Evader Motion Estimator (Cont.)

- **On-line Least Square: Approximated**

$$\theta_x^{(n+1)} = \theta_x^{(n)} + \rho(p_x - h^T\theta_x^{(n)})h$$
$$\theta_y^{(n+1)} = \theta_y^{(n)} + \rho(p_y - h^T\theta_y^{(n)})h$$

  - **Complexity:** only sums and multiplications

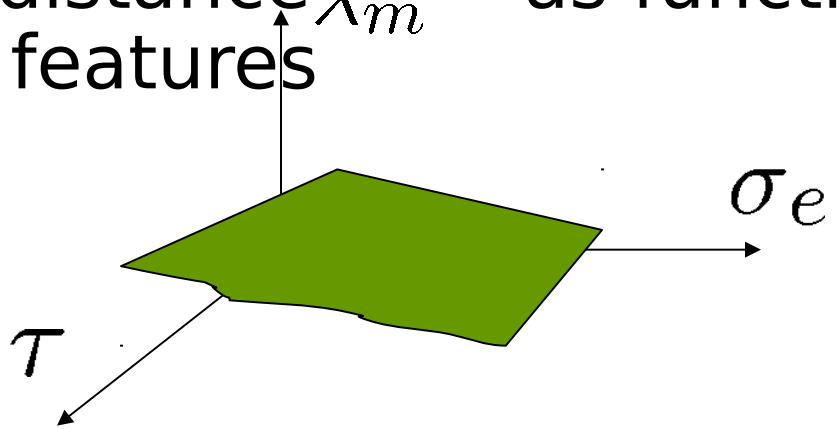
  - **Error bounds** on estimated parameter $(\sigma_{b_x}, \sigma_{v_x^e}, \sigma_{b_y}, \sigma_{v_y^e})$ are function of (b_x, v_x^e, b_y, v_y^e) H_n ρ

Roadmap:

- Compute optima $H_n \quad \rho \quad$ as a function o σ_s and n
- Compute $(\sigma_{b_x}, \sigma_{v_x^e}, \sigma_{b_y}, \sigma_{v_y^e})$ as a function o $H_n \quad \rho$
- Perform simulations to verify estimates
- Design controllers for mobile robots and for pan-and-tilt cameras
- Deploy battlefield of MICA nodes
- Implement algorithms on real setting

Future Work

- Find evader motion estimator and pursuer controller T_c
- Estimate capture time λ_m and mean evader-pursuer distance λ_m as function of the network features



- Use this map to estimate density of nodes and middleware specifics

Future work cont.

- Generalize algorithm to deal with smart evader
- Adopt a more accurate model for pursuer’s dynamics
- Tracking of multiple evaders

Distributed Map Building Using Multiple Mobile Robots

- **Process of establishing a representation of a previously unknown environment**

- **Examples of applications**
  - **A room or a hallway in a building with obstacles**
  - **A battlefield**
  - **An unknown terrain**
  - **Mars**

Relation with Other Applications

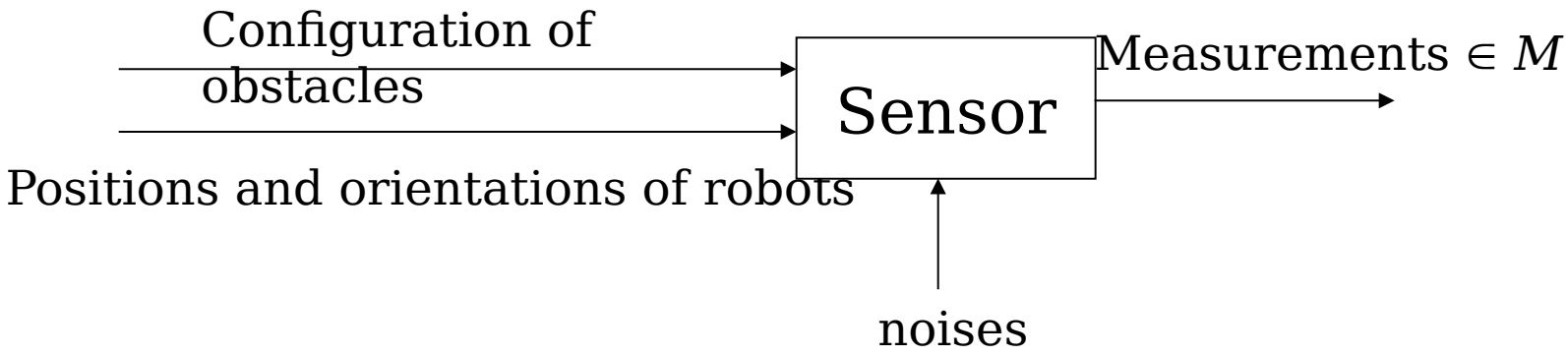
- **Front end: localization**
  - **In order to build a map, the robots should know where they are, at least approximately**
  - **SLAM (Simultaneously Localization And Mapping) problem**
- **Applications using the map to be built**
  - **pursuit-evasion game**
  - **art gallery problem**
  - **path planning and optimization, etc.**
- **Other related applications**
  - **Tracking (the obstacles to be mapped are mobile)**
  - **Environment monitoring**

Two Representations of Maps

- **How to represent the knowledge of an environment**
- **Geometric approach**
  - **Assume the shapes of the obstacles are predetermined (e.g. polygon, rectilinear polygon).**
  - **A map is a finite list of parameters characterizing these shapes.**
- **Occupancy grid approach**
  - **Partition the workspace into a set of grids *D*.**
  - **Associate with each grid $i \in D$ a number $p_i \in [0,1]$ representing the probability that grid *i* is occupied by an obstacle.**
  - **A map is the collection $\{p_i \mid i \in D\}$.**

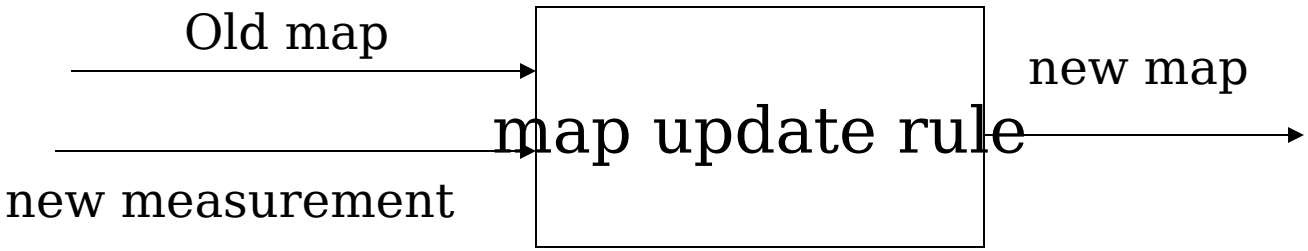
Sensor Model

- **Robots are equipped with sensors**
  - **For example, range sensors, touch sensors, and cameras**
- **The measurements take values in a certain set *M***
- **A sensor model is a collection of conditional probability distributions**



Map Update Rule

- **How to fuse new measurement with previous ones?**



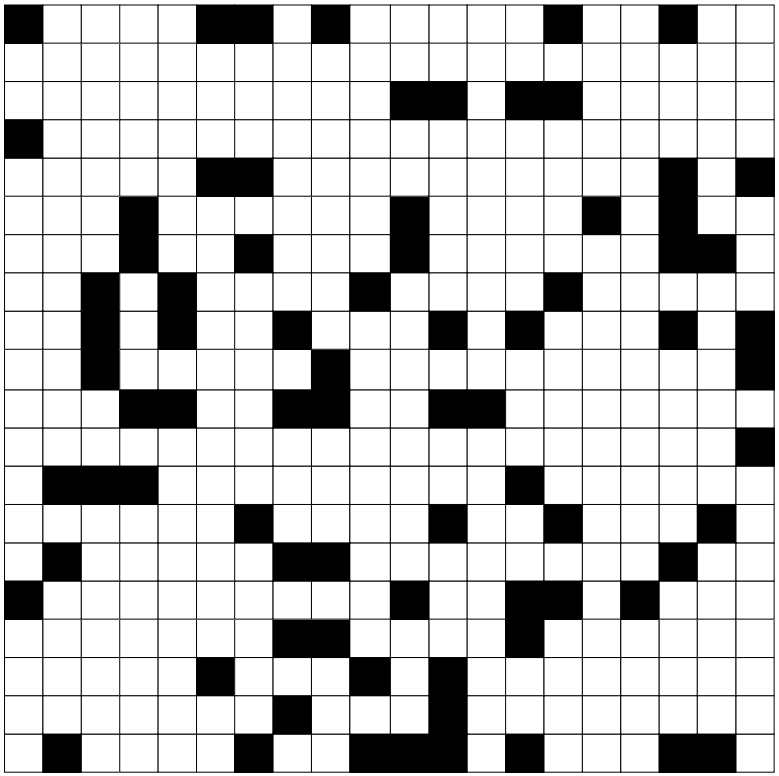
- **Bayes' Rule.**
- **Other inference rules.**

Model

- **The workspace is partitioned into rectangular grids.**
- **There are *k* mobile robots, whose states are specified by their positions *and* orientations.**
- **Each robot can move into four adjacent grids in one step.**
- **Each robot has the same sensor, and can measure two grids: the one it occupies and the one ahead of it.**
- **Sensor model is characterized by two probabilities**
  - **p_{00} : the probability of a grid measuring empty given it is empty**
  - **p_{11} : the probability of a grid measuring full given it is full**

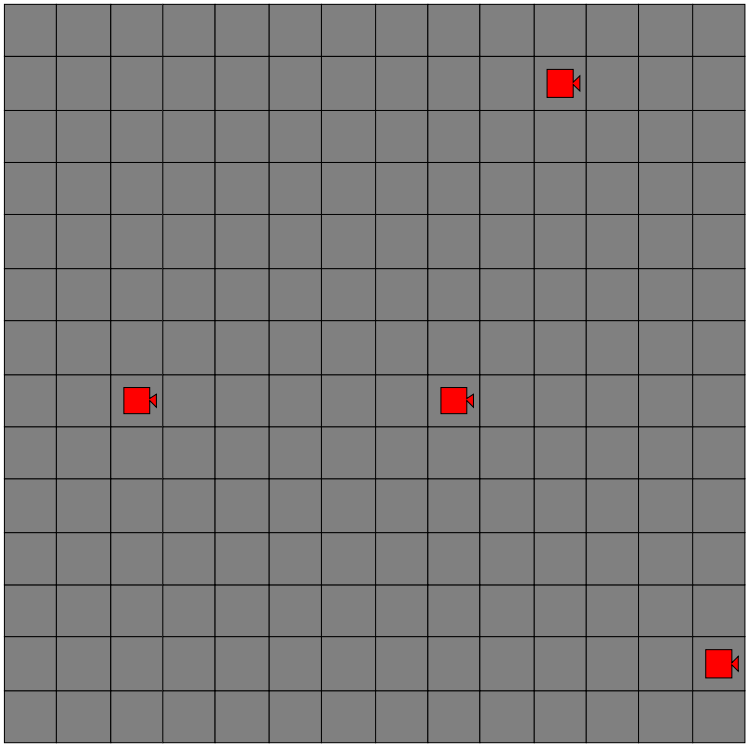
Generation of Obstacles

Obstacles are generated randomly, with each grid being occu
y an obstacle with probability p_f .



Initial Conditions

- Initial map is p_i=0.5, for all $i \in D$.
- Use grayness to indicate p_i , white: p_i=0; black: p_i=1.
- Initially the k robots are placed randomly in the workspac

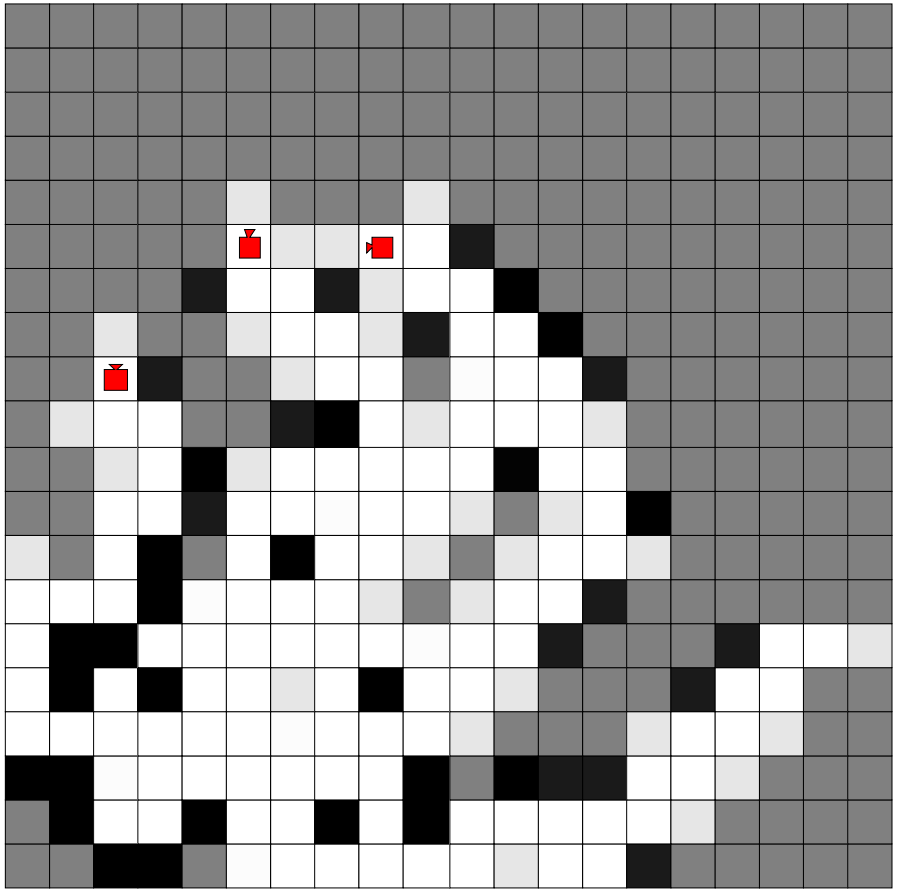


A Distributed Algorithm

- **At each time step**
  - **Robots take measurements from their current positions, and use these measurements to update the probabilistic map.**
  - **Partition the workspace into *Voronoi cells*;**
  - **For each robot, find within its cell the grid minimizing a weighted sum of its distance to the robot and a term reflecting the certainty of p_i in the current map, namely, $|\ln[p_i / (1- p_i)]|$.**
  - **Try to move the robot to an adjacent position closest to this grid.**
  - **If fail, switch the mode of the robot to *obstacle avoidance,* and try to turn the robot around the obstacle.**

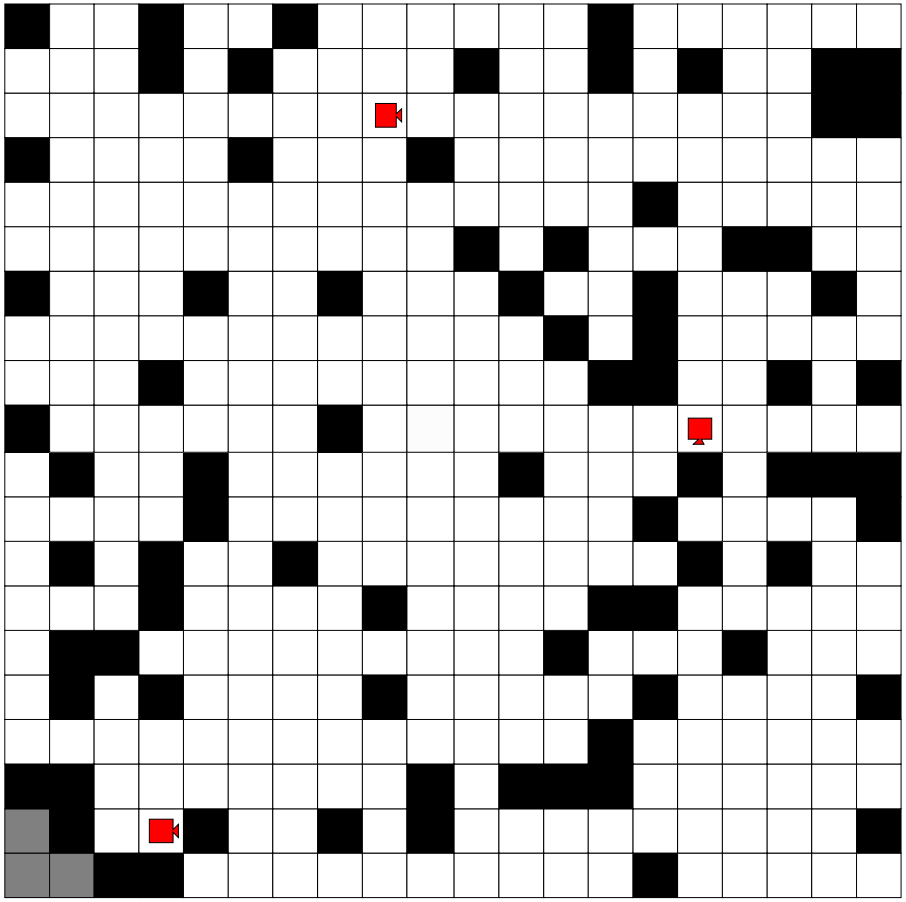
- **Repeat until certain stopping criteria are satisfied.**

Map Building in Progress...



Map Building Completed (or Stuck)

(because the workspace are not connected)



Current and future work

- **Tracking of moving objects:** determine the number and type of objects moving through a sensor field, and estimate the parameters of their trajectories.

- **Environmental monitoring:** detect a toxic plume and estimate the direction and speed of its wave front at any given point.

- **Distributed signal processing:** exploit a high level of correlation among signals in a massively distributed Sensorweb for more efficient coding and error correction.